Conformance Analysis of Execution Traces with Clinical Guidelines and Basic Medical Knowledge in Answer Set Programming*

Matteo Spiotta[1,2], Alessio Bottrighi[1], and Daniele Theseider Dupré[1]

[1] DISIT, Sezione di Informatica, Università del Piemonte Orientale
[2] Dipartimento di Informatica, Università di Torino

Abstract. Clinical Guidelines (CGs) are developed for specifying the "best" clinical procedures for specific clinical circumstances. However, a CG is executed on a specific patient, with her peculiarities, and in a specific context, with its limitations and constraints. Physicians have to use Basic Medical Knowledge (BMK) in order to adapt the general CG to each specific case, even if the interplay between CGs and the BMK can be very complex. In this paper, we focus on *a posteriori* analysis of conformance, intended as the adherence of an observed CG execution trace to CG and BMK knowledge. A CG description in the GLARE language is mapped to Answer Set Programming (ASP); the BMK and conformance rules are also represented in ASP, to perform conformance analysis, identifying non-adherence situations to CG and/or BMK, which must ultimately be evaluated by a physician in order to assess whether a trace can be considered as conformant or not.

1 Introduction

A Clinical Guideline (CG) is "a systematically developed statement to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances" [1]. The CGs are developed in order to capture medical evidence and to put it into practice, and deal with general classes of patients, since the CG developers (typically expert committees) cannot define all possible executions of a CG on any possible specific patient in any possible clinical condition. CG developers make some implicit assumptions: (1) *ideal patient*, i.e., patients have just the single disease considered in the CG (thus excluding the concurrent application of more than one CG), and are statistically relevant (they model the typical patient affected by the given disease), not presenting rare peculiarities or side-effects; (2) *ideal context*, i.e., in the context of execution, all necessary resources are available; (3) *ideal physicians* are executing the CG, i.e., physicians whose knowledge always allow them to properly apply the CGs to specific patients. On the other hand, when a CG is applied to a specific patient, the patient and/or the context may not be ideal. The physicians indeed exploit Basic Medical Knowledge (BMK) to adapt the CG to the specific case at hand.

* This research is partially supported by Compagnia di San Paolo.

The interplay between these two types of knowledge can be very complex, e.g., actions recommended by a CG could be prohibited by the BMK, or a CG could force some actions despite the BMK discourages them. Thus the physician judgment is very important in order to have a correct execution of a given CG in a specific case, as observed by the Infectious Diseases Society of America in its *Guide to Development of Practice Guidelines* [2]: "Practice guidelines, however, are never a substitute for clinical judgment. Clinical discretion is of the utmost importance in the application of a guideline to individual patients, because no guideline can ever be specific enough to be applied in all situations."

The issue of studying the interplay between the knowledge in CGs and BMK is relatively new in the literature. Several approaches have focused either on CGs or BMK in isolation, or have considered BMK only as a source of information, such as definitions of clinical terms and abstractions [3]. Only recently some approaches (e.g., [4, 5]) have considered that CGs cannot be interpreted and executed in "isolation", since CGs correspond to just a part of the medical knowledge that physicians have to take into account when treating patients. In this paper, we explore the interaction between CGs and BMK from the viewpoint of conformance analysis, intended as the adherence of an observed CG execution trace to both types of knowledge. Observe that both CG knowledge and BMK can be defeated (for a more detailed discussion see [4]), and it is the physician's responsibility to assess if a trace can be deemed as conformant or not. Our goal is to support the physicians in the conformance analysis task, providing them as much information as possible to make this task easier. The approach is based on GLARE ([6] and section 2) to represent CGs; our general framework is described in section 3 and its representation in Answer Set Programming in section 4. In particular, we provide a set of rules defining, on the one hand, discrepancies from one source of knowledge that are, at least potentially, justified by another source; on the other hand, discrepancies that are not justified.

2 GLARE representation formalism

In this section, we highlight some of the main features of the GLARE representation formalism (a detailed description is provided in [6]). GLARE distinguishes between *atomic* and *composite* actions. Atomic actions are elementary steps in a CG, in the sense that they do not need a further de-composition into sub-actions to be executed. Composite actions are instead composed by other (atomic or composite) actions. GLARE provides four different types of atomic actions:

- *work actions*, i.e., actions to be executed at a given point of the CG;
- *decision actions*, used to model the selection among alternative paths in a CG. GLARE provides *diagnostic* decisions, used to make explicit the identification of the disease the patient is suffering from, among a set of possible diseases, compatible with her findings. Such a decision is based on patient's parameters. GLARE also provides *therapeutic* decisions, used to represent the choice between therapeutic paths in a CG, based on a pre-defined set of parameters: effectiveness, cost, side effects, compliance and duration;

– *query actions* models a requests of information (typically patients' parameters), that can be obtained from the outside world (e.g. physicians, databases, patients visits or interviews). CG execution cannot proceed until such information has been obtained;
– *conclusion actions* represent the explicit output of a decision process.

Actions in a CG are connected through control relations. Such relations establish which actions might be executed next, in which order. GLARE introduces four different types of control relations: *sequence*, *concurrency*, *alternative* and *repetition*. The sequence relation explicitly establishes which is the next action to be executed; the alternative relation describes which alternative paths stem from a decision action, the concurrency relation between two actions states that they can be executed in any order, or also in parallel and the repetition relation, states that an action has to be repeated several times (i.e. the number of repetitions can be fixed a priori, or, alternatively, it can be asserted that the action must be repeated until a certain exit condition becomes true).

3 General Framework

A main goal of the framework presented in this paper is to exploit reusability of knowledge, in several ways:

– A model of the CG in Answer Set Programming (ASP, [7]) is derived automatically from the description of the CG in GLARE, and can be used for conformance analysis, as in this paper, i.e., analyzing if and how a single execution deviates from the CG, as well as for verifying properties of the CG, that should hold for all executions, using the approach in [8] as model checker in the loosely coupled framework in [9].
– A common repository of Basic Medical Knowledge can be used, in the framework in this paper, with models of different CGs.
– In perspective, an ontology of medical terms can provide the link for triggering BMK rules for a specific CG and its execution on a specific patient.

Figure 1 presents the general structure of the framework. The main entities, which are input to an ASP solver, are: the log, the CG model, the BMK, and a set of compliance annotation rules. The framework evaluates discrepancies between the log (actual execution) with the executions suggested by the CG, with the possible "variations" suggested by the BMK.

The log contains the data recorded during guideline execution. It includes data specific to the individual patient, such as medical records (from the Electronic Health Record, EHR) and the actions performed on the patient; it also includes data related to the context (e.g., hospital) in which the CG is performed, such as availability of equipment and personnel.

The ASP model of the CG encodes all the admitted treatment paths provided by the CG. Tools such as GLARE provide a formal representation of CGs, which can be translated to ASP. In this framework, information on when an action is

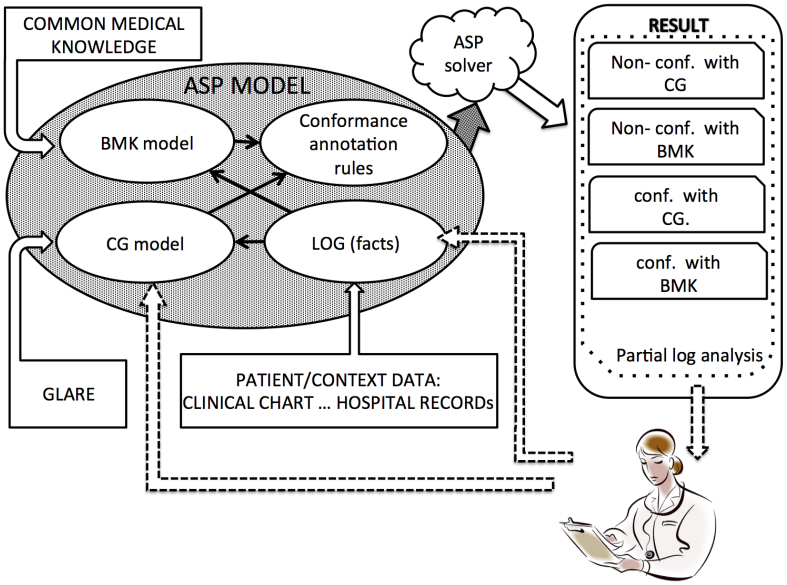

Fig. 1. General framework

executed is used both to verify whether it is justified by the CG, and to justify execution of subsequent actions in the CG. Both the control flow perspective and the data perspective of the GLARE CG specification is encoded in the CG ASP model. In the current version, quantitative time constraints in GLARE are not supported.

To better evaluate the interplay of BMK and CG we take in account the execution model of actions shown in figure 2, similar to the one in [4]. At a given point in the execution of a CG on a specific patient, the control relations in the CG or rules in the BMK indicate that a given action have to be executed (is a candidate). A candidate action is discarded if its preconditions (modeled in the CG) are false; or it may be discarded because of conditions not explicitly modeled in the CG; we expect that some of such reasons for discarding are modeled in the BMK. Decision and conclusion actions are instantaneous and, once started, they can be considered concluded. Work actions and query actions, once started, can either be completed or aborted. An action is aborted if a failure occurs during its execution, or it may be aborted because some condition arises; again, we expect that some of such reasons for aborting are modeled in the BMK.

Once an action of the CG is discarded or aborted, in general we cannot infer the correct way to continue the execution of the CG. In some cases the physician would continue the execution skipping the uncompleted action (e.g., for an action having minor impact on the treatment), cases in which she would restart the execution from some point further away (e.g., a previous decision point or the end of the partial plan); in other cases, the entire CG should be interrupted (e.g., the action is essential for the treatment). We do not assume that this information

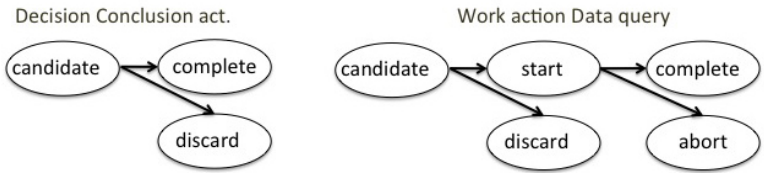

Fig. 2. Action model

is modeled, therefore we support interaction of the framework with the analyst which will suggest where in the CG and in the log the analysis can be restarted.

The annotation compliance rules are the keystone of the entire framework. They define the output of the analysis, and are triggered by discrepancies, starting from the actions recorded in the log and the expected actions derived from the CG and BMK. Two different classes of discrepancies are provided:

- Discrepancies of the log with a knowledge source (CG or BMK rule) which are "supported" by another source.
- Discrepancies of the log with a knowledge source (KS) not supported by other knowledge.

While the second class represents incorrect behavior (wrt the considered KSs), the first one represents a case of (at least, potential) conflict between knowledge sources. Which one should prevail cannot be stated in general [4], and providing knowledge for stating it for all cases is, in general, too costly. Therefore we provide the information in the log, which can be filtered further by the analyst.

We assume completeness and correctness of the Log. Completeness with respect to actions means that for all actions taken, the following is recorded:

- start, discard, abort, complete and failure reason (human and/or technical problem which caused incorrect completion of an action);
- the outcome of completed decision actions.

Completeness with respect to (patient or context) data means that it contains record of data which have driven the control flow (CF) and data which could force the physician to change the normal execution applying BMK rules.

Correctness means that only verified information is recorded, no conflicting data can be stored (e.g., an action is first discarded and then completed).

We expect (see [4]) that the BMK provides pieces of knowledge such as:

- Actions of a given type, or specific actions, are contraindicated for patients in a given temporary or permanent conditions; e.g.:
  - treatment with a drug D is contraindicated for patients (known to be) allergic to D;
  - an invasive exam or therapy is contraindicated in some cases.
- the execution of a CG may (have to be) suspended if a life threat arises, and the latter should be treated. Whether the execution actually *has* to be suspended depends, in general, on whether the current actions being

executed are compatible with the treatment of the life threat, and the life threat itself. Specific knowledge in this respect may be available or not. We intend that the life threatening problem, e.g., a heart failure, is not part of the class of problems dealt with by the CG, i.e., in the example, the CG currently being executed is not the one for cardiovascular diseases. The source of knowledge for its treatment should, in principle, come from another CG; however, in this paper we do not address the problem of interaction of multiple CGs and we assume to have available, when analyzing logs for the execution of a CG, the set of possible treatments for other problems.
– Actions of a given type (e.g., routine exams) can be performed even if not part of the CG.

4 ASP representation and conformance rules

In this section we describe the ASP representation of the Log, CG model, BMK model, and the annotation rules.

In the Log, context and patient data, decision outcomes, and action states (discard, started, aborted, completed) are encoded as facts associated with a timestamp. Based on the set of facts *data(name,value,timeStamp)*, *action(actID, actState,timeStamp)*, *decided(actID,actIDoutcome,timeStamp)*, we reconstruct the time line for the framework by means of the predicate *next*, as follows:

```
next(S,SN):-state(S),state(SN),SN>S, not stateinbetween(S,SN).
stateinbetween(S,S2):-state(S),state(S2),state(S3),S<S3,S3<S2.
```

A predicate *state(S)* is true for all timestamps S; `next(S,SN)` is true for all the pairs of timestamps with no fact in between. Predicate `holds(var(d,c),S)` represents the value v of data d in state S. The rules below propagate data values up to the next change:

```
holds(var(N,V),SN):- holds(var(N,V),S), next(S,SN),
                not holds(var(N,V1),SN), V!=V2, data(N,V1,_).
holds(var(N,V),S):-data(N,V,S).
```

The CG model is not reported in full detail. The main CG component is the control flow (CF), we encode it in ASP with an approach similar to the one in [8]. The CF model defines `candidate(A,S)` for actions A and states S. There are atomic actions and composed actions. Predicates `end(type,ID,S)` and `start(type,ID,S)` are defined to reconstruct the execution interval of composite actions from the action states of atomic actions, registered in the log (completed/started). The definition of *end* relates the end of atomic actions to the end of composite actions and control structures; e.g., a set of concurrent actions is considered ended in S only if all the sub-processes are ended in S.

Every candidate action a (atomic/composite) can be executed, and once it ends, it enables its CF successors *a1* in the *next* time state by means of the predicate `candidate(A,SN)`:

```
(a) candidate(A1,SN):-succ(A,A1),not excp(A1,S),end(_,A,S),next(S,SN).
```

```
(b) candidate(A1,SN):-decided(A,A1,S),end(action,A,S),next(S,SN).
(c) candidate(A1,SN):-end(_,A,S),next(S,SN),reExecute(A,S).
```

In rule (a), A1 is candidate in SN if A ended in S and A1 is the *successor* of A in the flow. Predicate `excp(A,S)`, used also in rules defining *end*, blocks execution after actions terminated with errors or not admitted by the CF (e.g. a completed data query without data, an action executed but not candidate). Rule (b) encodes *decisions*: the predicate *decided*, the outcome of the decision task registered in the log, enables the successor chosen by the physician. Rule (c) encodes *repetition*. All actions (atomic/composite), if specified by the CG, can be re-executed: the predicate `reExecute(ID,state)` is true if the action ends and the exit condition on data are false. Other CG specifications mapped in ASP are the list of data requested by a data query action, parameters to evaluate therapeutic decision, exit conditions for repetition and precondition of action.

The BMK model consists in a set of rules which prescribe or allow the introduction or cancellation of an action, based on conditions on the patient and contextual data. Such *conditions* are defined in other rules.

```
prescribe(id,A,normal/urgent,S):- condition(S).
allow(id,A,S):- condition(S).
prescribeCanc(id,A,S):-condition(S).
allowCanc(id,A,S):- condition(S).
```

The *id* is used to point out in the analysis the rule that has generated or justified a discrepancy. Multiple instances of *prescribe* with the same id and different actions encode the request to execute one of a set of possible alternative actions. The third argument (*normal*/*urgent*) encodes the urgency to execute the action. In the *normal* case, there is no constraint on the order of execution wrt other actions, while, in the *urgent* case, it must be the first action to be executed once the condition is true. The difference between *prescribe* and *allow* is that in the first case a discrepancy is reported (annotated in different ways) both when the action is executed or not, in the second case we report a discrepancy only when the rule is "applied" (according to the log, A is discarded or aborted, in case of `allowCanc(_,A,_)`, A is candidate in case of `allow(_,A,_)`).

The four predicates are related to action events as follows:

```
candidate(A,S,ID):-prescribe(ID,A,normal,S),not running(A,S).
candidate(A,S,ID):-prescribe(ID,A,urgent,S),not running(A,S).
urgent(ID,A,S):-prescribe(ID,A,urgent,S),not running(A,S).
candidate(A,S,ID):-allow(ID,A,S),action(A,started,S).
discard(A,S,ID):-prescribeCanc(ID,A,S),candidate(A,S).
abort(A,S,ID):-prescribeCanc(ID,A,S),running(A,S).
discard(A,S,ID):-allowCanc(ID,A,S),action(A,discarded,S),candidate(A,S).
abort(A,S,ID):-allowCanc(ID,A,S),abort(A,S,ID).
```

Predicates *candidate*, *discard* and *abort* are used in the annotation rules to point out discrepancies.

In the following we provide the representation for the BMK examples in [4].

BMK: The execution of any CG may be suspended, if a problem threatening the patients life suddenly arise. Such a problem has to be treated first. An immediate response for acute heart failure could be a Diuretic Therapy.
An encoding for such knowledge is:

```
lifeThreat(heartFailure). lifeThreat(stroke). [...]
treatment(heartFailure,diureticTherapy).
treatment(heartFailure,betaBlockerTherapy).
treatment(heartFailure,inotropeTherapy) [...]
prescribeCanc(r1,T,S):-holds(var(D,true),S),lifeThreat(D),
          running(T,S),not treatment(D,T).
prescribe(r1,T,urgent,S):-treatment(D,T),holds(var(D,true),S),
          lifeThreat(D),not running(T2,S),treatment(D,T2).
```

BMK: Calcemia and glycemia are routinely performed in all patients admitted to the internal medicine ward of Italian hospitals, regardless of the disease.

```
routineAct(glycemia). routineAct(glycemia).
allow(r2,A,S):-action(adm_internal_medicine,started,S),routineAct(A).
allow(r2,A,S):-allow(A,S),action(A,started,S),next(S,SN),routineAct(A).
```

BMK: Contrast media administration for coronary angiography may cause a further final deterioration of the renal functions, in patients affected by affected by unstable advanced predialytic renal failure.

```
prescribeCanc(A,S,bmk1):- contraindicated(A,S).
contraindicated (angiography,S):-state(S),
          holds(var(advanced_predialytic_renal_failure,true),S).
```

Conformance annotation rules are as follows. In a state t, relatively to action a, the following discrepancies, potentially justified by a KS, are defined:

A1 **A discrepancy with the CG, justified by a BMK rule *r***, if a is recorded as discarded in t, rule r prescribes discarding a, and the CG suggest a as candidate.

A2 **A discrepancy with a BMK rule *r*, justified by the CG**, if a is recorded as started in t, rule r prescribes discarding a, and the CG suggest a as candidate.

A3 **A discrepancy with a KS *s*, justified by BMK rule *r***, if a is recorded as aborted in t, rule r prescribes aborting a and, until t, action a was running as suggested by s.

A4 **A discrepancy with a BMK rule *r* justified by the KS *s***, if a is recorded as completed in t, rule r prescribes aborting a and, until t, action a was running as suggested by s.

A5 **A discrepancy with a BMK rule *r* justified by the KS *s***, if in a state in t a rule r prescribes with urgency one or more actions and a different action, prescribed by s, is recorded as started.

In a state t, relatively to action a, the following discrepancies not justified by other knowledge sources are output:

B1 **A discrepancy with the KS *s***, if a is recorded as discarded in t, s suggests a as candidate, no BMK rule r justifies discarding a and preconditions of a are satisfied at t.

B2 **A discrepancy with the KS *s***, if a is recorded as started in t, s suggests a as candidate and preconditions of a are falsified at t.

B3 **A discrepancy with all KSs**, if a is recorded as started in t, there is no source s which suggest a as candidate.

B4 **A discrepancy with the KS *s***, if a is recorded as aborted, no BMK rule r justify aborting a, no failure is recorded for a and, until t, action a was running as suggested by s.

B5 **A discrepancy with the CG**, if a was candidate by the CG at t and, after t, a is not recorded as started.

B6 **A discrepancy with a BMK rule *r***, if in an interval $[t_0, t]$ a rule r suggest a, and possibly alternative actions, as candidates; the actions are not candidate at *t+1*, and at *t+1* no action suggested by r is executed.

B7 **A discrepancy with the CG**, if action a is recorded as completed, a is the successor of a *decision* action, and a is not eligible, given data at time t.

B8 **A discrepancy with the CG**, if action a is recorded as completed, a is a *query* even though not all necessary data were present in the log at time t.

The encoding of the rules in ASP is relatively straightforward. E.g., for A1 and A2 we have the two pairs of clauses below:

```
discrepancy(cg,ID,A,S):-action(A,discarded,S),discard(A,S,ID),
            not precondFalse(A,S),candidate(A,S).
discrepancy(ID2,ID,A,S):-action(A,discarded,S),discard(A,S,ID),
            not precondFalse(A,S),candidate(A,S,ID2).
discrepancy(ID,ID2,A,S):-action(A,started,S),discard(A,S,ID),
            candidate(A,S,ID2).
discrepancy(cg,ID,A,S):-action(A,started,S),discard(A,S,ID),candidate(A,S).
```

An ASP solver such as Clingo [10] computes an answer set of the overall ASP model (see figure 1); the set of instances of the *discrepancy* predicate in the answer set contains the information necessary to produce a user-friendly result.

5 Conclusions

We presented a framework for analyzing conformance of execution traces for patient treatment with Clinical Guidelines and Basic Medical Knowledge.

The approach, as presented in the paper, is specific to healthcare processes, but a similar one can be used for comparing actual execution traces with process models in other organizations, i.e., for *business* processes; also in other contexts, in fact, there might be "ideal" process models, which make sense as a reference, but do not define all conceivable process adaptations in all situations.

Conformance analysis work in the process model area, e.g. [11–13], is mainly devoted to measuring the adherence of a model with execution traces, in order

to refine a model, rather than to analyze, as in our approach, the correctness of an execution with respect to a model.

Our approach is quite similar to [4], which is mainly devoted to the interaction between CG and BMK, while this paper is focused on the identification and classification of non-adherence situations to CG and/or BMK.

As regards CGs and medical knowledge, the approach does not take into account the general problem of interaction of multiple CGs, where general medical knowledge should of course play a role (in particular, models of interactions of different diseases and different drugs). Also, the framework does not explicitly address the extraction of part of the BMK from available medical ontologies.

References

1. Field MJ and Lohr KN, editors. *Guidelines for clinical practice: from development to use*. National Academy Press, Institute of Medicine, Washington, D.C, 1992.
2. M. A. Kish. Guide to development of practice guidelines. *Clinical Infectious Diseases*, 32(6):851–854, 2001.
3. A. Ten Teije, S. Miksch, and P. Lucas, editors. *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*, volume 139 of *Studies in Health Technology and Informatics*, Amsterdam, 2008. IOS Press.
4. A. Bottrighi, F. Chesani, P. Mello, M. Montali, S. Montani, and P. Terenziani. Conformance checking of executed clinical guidelines in presence of basic medical knowledge. In F. Daniel, K. Barkaoui, and S. Dustdar, editors, *Business Process Management Workshops (2)*, volume 100 of *LNBIP*, pages 200–211. Springer, 2011.
5. C. J. Brandhorst, D. Sent, R. A. Stegwee, and B. M. A. G. van Dijk. Medintel: Decision support for general practitioners: A case study. In K.-P. Adlassnig, B. Blobel, J. Mantas, and I. Masic, editors, *MIE*, volume 150 of *Studies in Health Technology and Informatics*, pages 688–692. IOS Press, 2009.
6. P. Terenziani, G. Molino, and M. Torchio. A modular approach for representing and executing clinical guidelines. *Artificial Intelligence in Medicine*, 23(3):249–276, 2001.
7. M. Gelfond. Answer Sets. *Handbook of Knowledge Representation, chapter 7, Elsevier*, 2007.
8. L. Giordano, A. Martelli, M. Spiotta, and D. Theseider Dupré. Business process verification with constraint temporal answer set programming. *Theory and Practice of Logic Programming*, 13(4-5), 2013 (online).
9. A. Bottrighi, L. Giordano, G. Molino, S. Montani, P. Terenziani, and M. Torchio. Adopting model checking techniques for clinical guidelines verification. *Artificial Intelligence in Medicine*, 48(1):1–19, 2010.
10. M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam answer set solving collection. *AI Comm.*, 24(2):105–124, 2011.
11. A. Adriansyah, B.F. Dongen, and W.M.P. Aalst. Towards robust conformance checking. In M.l Muehlen and J. Su, editors, *Business Process Management Workshops*, volume 66 of *LNBIP*, pages 122–133. Springer, 2011.
12. A. Rozinat and W. M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, March 2008.
13. Jorge Munoz-Gama and Josep Carmona. Enhancing precision in process conformance: Stability, confidence and severity. In *Proc. of CIDM*, pages 184–191, 2011.